Noise-contrastive estimation of normalising constants and GANs

Fonctions génériques

Algorithme d'Hasting

Utilité : simuler selon $p_m(.,\psi)$ pour un paramètre ψ choisi.

Argument	Type	Exemple	Indication
x	vecteur	rcauchy(100, 0, 1)	notre échantillon de densité inconnue
n	entier	100	taille de la simulation
psi	vecteur	c(0,1)	paramètres de la fonction h
h	fonction		fonction qui retourne $\overline{p_m}(.,\psi)$

```
hasting = function(x, n, psi, h){
  y = c()
  y = append(y, sample(sample(x, 1)))
  for (i in 2:n){
    y_ = rnorm(1, y[i-1], 1)
    u = runif(1)
    if ( u <=
         (h(y_,psi) * dnorm(y_, y[i-1], 1))
         /(h(y[i-1],psi) * dnorm(y[i-1], y_, 1))
    ){
      y = append(y, y_)}
    else {
      y = append(y, y[i-1])
    }
  }
  return (y)
}
```

Note : on peut très certainement écrire sous forme matricielle cette fonction pour une meilleure performance.

MC MLE

Utilité : retourne une estimation des paramètres selon la méthode décrite dans le papier de Geyer.

```
mc_mle = function(x, n, psi, h){

  y = hasting(x, n, psi, h)

  L = function(theta){
    return(sum(log(h(x,theta)/h(x,psi))) - n*log(mean(h(y,theta)/h(y,psi))))
  }

  theta = optim(
    par = rep(1,length(psi)),
    gr = "CG",
```

```
    control = list(fnscale=-1),
    fn = L
  )$par

  return(theta)
}
```

NCE

Utilité : Retourne l'estimation de la constante et des paramètres.

Argument	Type	Exemple	Indication
x	vecteur	rcauchy(100, 0, 1)	notre échantillon de densité inconnue
law_y	fonction	rnorm	fonction qui retourne un échantillon suivant la loi p_n
n	entier	100	taille de l'échantillon de bruit suivant la loi p_n
params_y	vecteur	c(0,1)	arguments de la fonction law_y
log_pm	fonction		fonction qui retourne le logarithme de la densité p_m
log_pn	fonction		fonction qui retourne le logarithme de la densité p_n
size_theta	entier	3	taille de θ, vaut habituellement 2 ou 3
method	string	"CG"	méthode d'optimisation, habituellement "CG" ou "BFGS"

```
nce = function(x, law_y, n, params_y, log_pm, log_pn, size_theta, methode = "CG"){

  y = do.call(law_y,c(list(n),params_y))

  m = length(x)

  h = function(u, theta){
    return( 1 / (1 + n/m * exp(log_pn(u) - log_pm(u, theta))))
  }

  J = function(theta){
    return( sum(log(h(x, theta))) + sum(log(1 - h(y, theta))) )
  }

  theta = optim(
    par = rep(1, size_theta),
    gr = methode,
    control = list(fnscale=-1),
    fn = J
  )$par

  return(c(theta[-size_theta], exp(-theta[size_theta])))
}
```

Graphiques

```
library(ggplot2)
library(reshape)
```

Utilité : afficher l'histogramme pour un échantillon de données x.

```
print_hist = function(x) {

  df = data.frame(x = x)

  hist_x = ggplot(df, aes(x=x)) + geom_histogram(aes(y = stat(count) / sum(count)), bins = 20, color="wh

  print(hist_x)
}
```

Utilité : afficher l'évolution des paramètres au fur et à mesure de l'augmentation de n (la dimension de l'échantillon de bruit)

```
evolution_paramètres = function(x, law_y, params_y, log_pm, log_pn, nb_of_params, taille, precision, lab

  # Creation de l'abscisse
  m = length(x)
  N = seq(10, m*taille, length.out = precision)

  # Creation de l'ordonnée
  theta = c()
  for (n in N) {
    theta = append(theta, nce(x, law_y, n, params_y, log_pm, log_pn, nb_of_params, methode))
  }

  # Formatage des données
  theta = t(rbind(matrix(theta, nrow = nb_of_params),N))
  df = as.data.frame(theta)
  df_melted = melt(df, id.vars = 'N')

  # Plot
  plot_df = ggplot(df_melted, aes(x = N, y = value)) +
  geom_line(aes(color = variable, group = variable)) +
  geom_point(aes(color = variable, group = variable)) +
  labs(title = "Evolution des paramètres", x = "n", y = "Paramètres", color = "Légende") +
  scale_color_manual(labels = labels, values = c("blue", "red", "orange"))

  print(plot_df)

  return(theta)
}
```

Exemple basique : la loi normale

Soit x l'échantillon de taille m obtenu selon la loi de densité inconnue p_d.

On considère ici que p_d appartient à la famille de fonctions paramétrées par $\theta = (c, \mu, \sigma)$ suivante :

$$p_m(u;\theta) = \frac{1}{Z(\mu,\sigma)} \times exp\big[-\frac{1}{2}\big(\frac{u-\mu}{\sigma}\big)^2\big] \quad \text{d'où} \quad ln(p_m(u;\theta)) = c - \frac{1}{2}\big(\frac{u}{\sigma} - \frac{\mu}{\sigma}\big)^2$$

```
pm_barre = function(u, theta){
  return(exp(-0.5 * ((u - theta[1]) / theta[2]) ** 2))
}
```

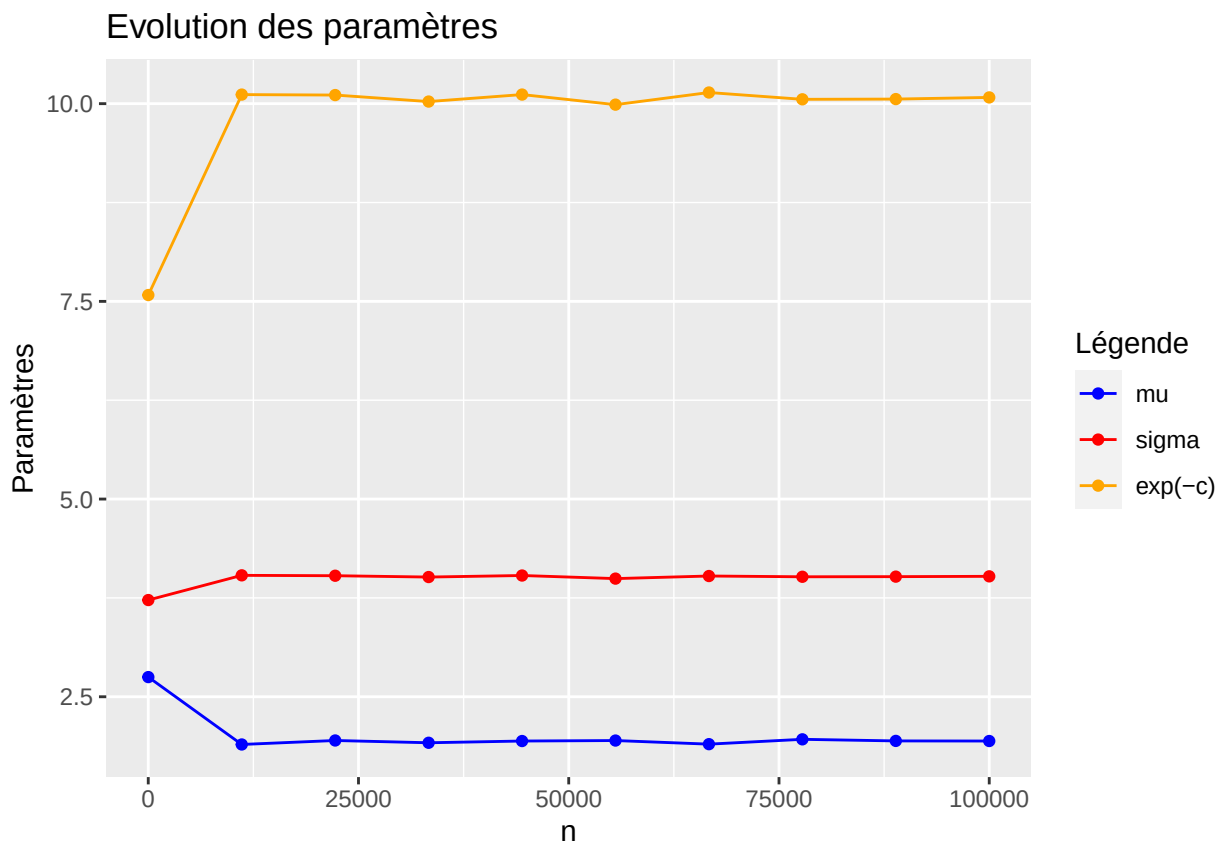
```r
log_pm = function(u,theta){
  return(theta[3] - 1/2 * (u/theta[2] - theta[1]/theta[2]) ** 2)
  # theta[1] = mu / theta[2] = sigma / theta[3] = c
}

log_pn_cauchy = function(u){
  return(log(dcauchy(u, mean(x), sd(x))))
}

m = 10000
n = 100000
x = rnorm(m, 2, 4)
size_theta = 3

# METHODE MC MLE
mc_mle(x, m, c(mean(x),sd(x)), pm_barre)
```

```
## [1] 2.069748 4.030914
```

```r
# METHODE GEYER
nce(x, rcauchy, n, c(mean(x),sd(x)), log_pm, log_pn_cauchy, size_theta)
```

```
## [1]  1.945709  4.009502 10.063665
```

```r
evolution_paramètres(x, rcauchy, c(mean(x),sd(x)), log_pm, log_pn_cauchy, size_theta, 10, 10, c("mu", "
```




```
##                                 N
```

```
##  [1,] 2.748683 3.722803  7.579817     10
##  [2,] 1.896384 4.035345 10.114793  11120
##  [3,] 1.946828 4.030437 10.108781  22230
##  [4,] 1.918081 4.013684 10.026766  33340
##  [5,] 1.940023 4.033140 10.115145  44450
##  [6,] 1.946126 3.994001  9.987630  55560
##  [7,] 1.900566 4.026703 10.141069  66670
##  [8,] 1.961516 4.017153 10.055099  77780
##  [9,] 1.941497 4.018772 10.057914  88890
## [10,] 1.940066 4.022623 10.078969 100000
```